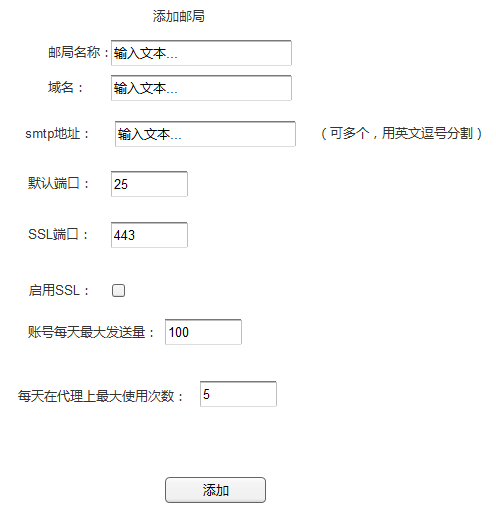
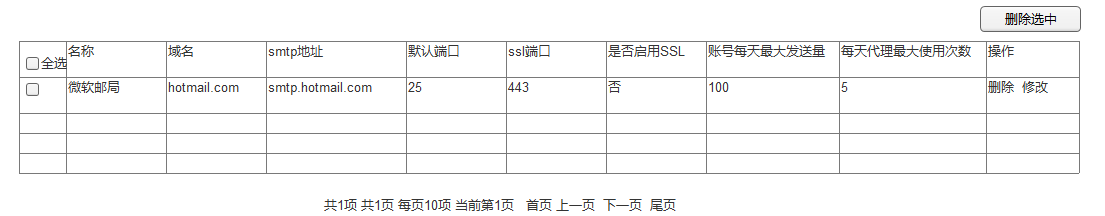
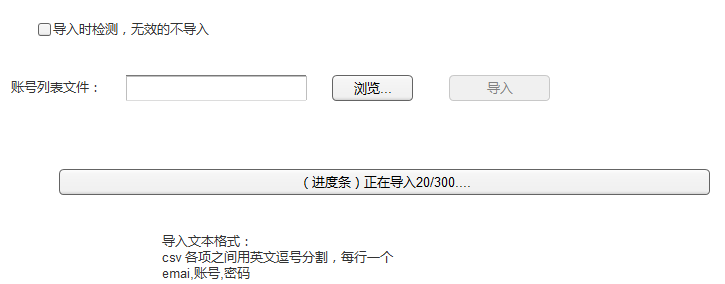
1. 邮局表
2. 邮局名称，
3. 域名，
4. smtp地址列表，
5. 默认端口，
6. 是否启用SSL，
7. 账号每天在代理上最大使用次数





1. 发送账号表
2. email账号、
3. 密码
4. 是否启用
5. 添加时间

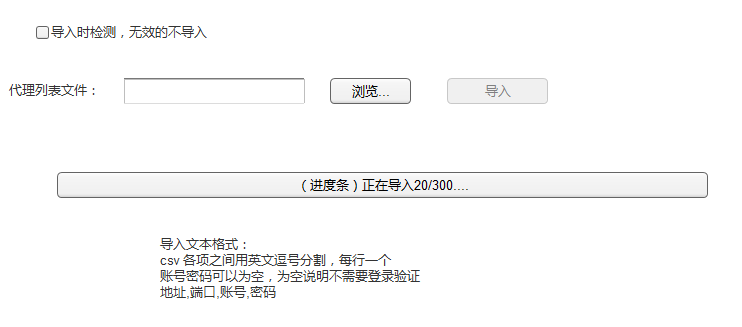


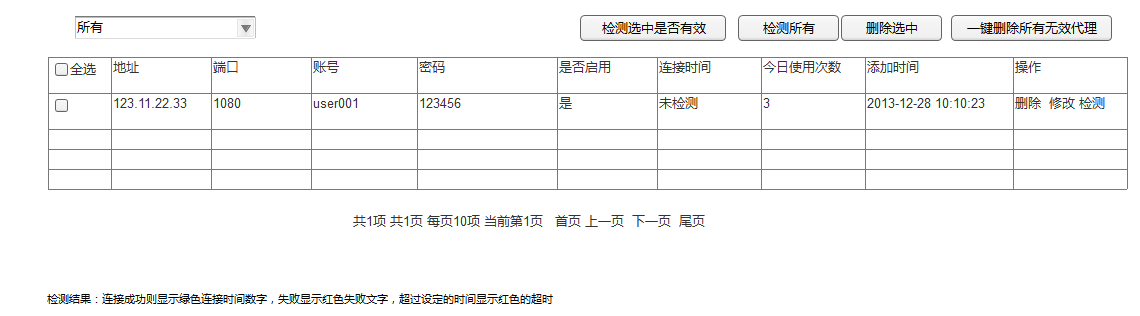




1. socks代理地址表
2. 地址（ip或域名），
3. 端口，
4. 账号，
5. 密码，
6. 是否启用，
7. 添加时间







1. 任务列表
2. 任务名称，
3. 数据路径，
4. 状态（未启动，发送中，暂停，完成），
5. 启动时间，
6. 完成时间，
7. 总数据量
8. 成功数量
9. 失败数量
10. Offer链接
11. 取消订阅链接
12. 投诉链接
13. Pixel代码
14. 发送数量=成功数量+失败数量
15. 剩余数量=总数量-发送数量

Task文件夹

任务要发送的数据保存在一个文件夹里，里面包含：

1、发件人（fromline）文件，一行一个发件人，发送时随机选择 fromlines.txt

2、标题（subjectline）文件，一行一个标题，发送时随机选择 subjectlines.txt

3、邮件正文文件夹：下面包含若干个文件，包含html和txt两种类型，html文件以html格式发送，txt文件以txt格式发送。发送时随机选择一个文件为正文内容，

Creatives文件夹 邮件正文文件夹

收件人列表文件包含收件人列表，一行一个，每行的数据各个列用英文逗号分割。

第一个是emmail地址，后面是扩展内容。

list.txt是收件人列表，一行一个，如果不包含扩展属性，一行就一个email,如果包含扩展属性，则用csv格式，即每行数据的各个列用英文逗号分割。第0个是email地址，后面是扩展内容。

例如：

adrian.bean@gmail.com,adrian,bean,f,25

adrian.becerra@gmail.com,adrian,bentley,m,30

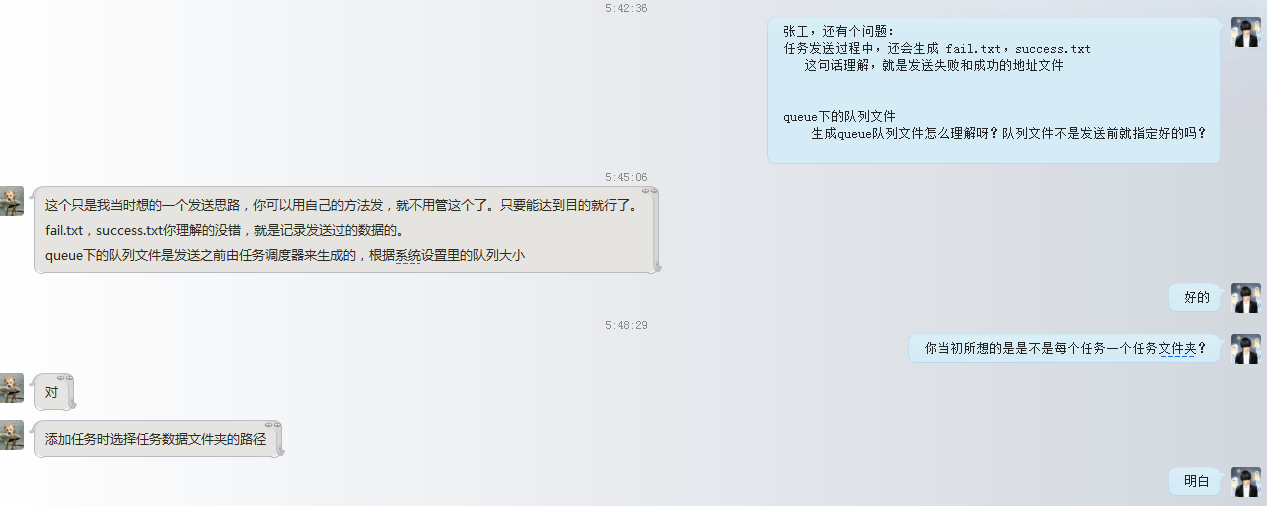
[adrian.bentley@gmail.com,adrian,bentley,m,35](mailto:adrian.bentley@gmail.com,adrian,bentley,m,35)

任务发送过程中，还会生成 fail.txt，success.txt和

queue下的队列文件

生成失败和成功txt，理解，生成queue队列文件，不理解

这个只是我当时想的一个发送思路，你可以用自己的方法发，就不用管这个了。只要能达到目的就行了。  
fail.txt，success.txt你理解的没错，就是记录发送过的数据的。  
queue下的队列文件是发送之前由任务调度器来生成的，根据系统设置里的队列大小



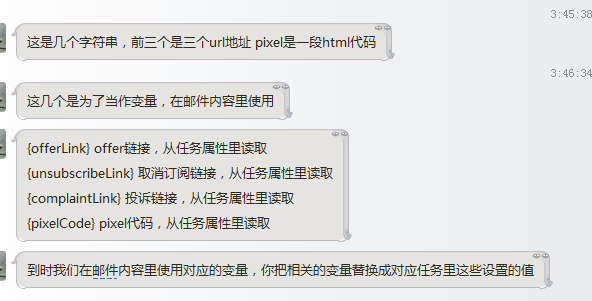
这个文件夹的内容具体可以看任务数据样板 文件夹

Email账号使用记录表

Email账号 日期 使用次数

Socks使用记录表

邮局域名，socks5地址 日期 使用次数







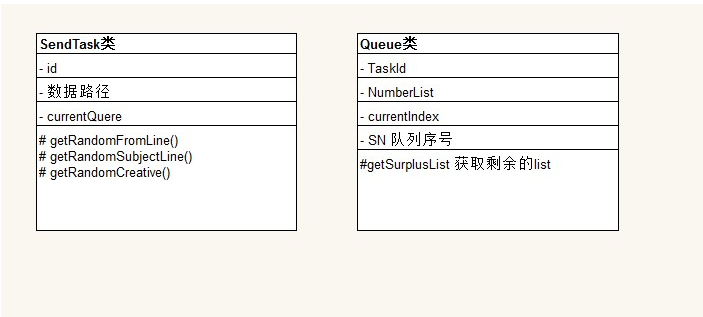
1. 系统设置



1. 模块
2. 错误日志记录

使用log4j，把程序异常日志记录下来，方便以后排错

1. 任务调度用到的两个类



1. 发送账号调度器

GetSendAccount(待发email)方法

锁定

获取一个【启用的&&有效的&&今天未发送够量的&&没有正在使用的&&和代发email同域名的账号】，获取到则修改使用状态为正在使用，返回。

未获取到则获取一个【所有启用的&&有效的&&今天未发送够量的&&没有正在使用的&&和代发email不同域名的账号】，获取到则修改使用状态为正在使用，返回。

以上都未获取到则返回null

Update(account)方法

{

如果发送成功，今天使用次数++

修改使用状态为未使用

}

1. 代理任务调度器

GetSocks(待发email)方法

锁定

获取一个【启用的&&有效的&&今天此socks未发送够量的&&没有正在使用的】，获取到则修改使用状态为正在使用，返回。未获取到则返回null

Update(代理地址)方法

{

如果发送成功，今天使用次数++

修改使用状态为未使用

}

1. 任务调度器

负责队列的创建：同一写入成功和失败的list，响应发送线程的请求，提供可以发送的任务，统一提供任务和发送队列的分配。

方法：getTask()获取一个可以正在发送的任务，如果队列文件为空且已经生成过队列，说明此任务已发送完毕，修改任务状态，并返回null，

否则new一个SenTask类，用来传递发送数据，把任务id和数据路径赋给它，给此任务加锁，判断是否已经生成队列，没有生成，获取最小的队列序号，根据队列序号从数据路径读取相应的队列文件，文件不存在则获取下一个。

获取队列内容后，删除队列文件，解除锁定，new 一个Queue类，用获取到的队列内容初始化，然后赋值给SendTask实例的currentQuere。返回SendTask实例。

Turnback(sendTask)

{

调用sendTask.currentQuere.getSurplusList()获取剩余list

根据sendTask.数据路径和sendTask.currentQuere.SN写入队列文件。

}

1. 发送任务处理流程

软件启动后，开启指定数量的线程。

每个发送任务线程的工作：

Var currentTask;

While(！请求关闭程序)

{

向任务调度器请求任务，请求不到则休息指定的时间。

有正在运行的任务时，任务调度器会返回来一个SendTask实例，

存到currentTask，然后开始遍历curentTask.currentQuere.NumberList.

Foreach(var email in currentQueue.currentQuere.NumberList)

{

向发送账号调度器请求发送账号，请求不到则休息指定时间后重试，获取到发送账号后，然后向代理调度请求代理，请求不到照样休息指定时间重试。

请求代理之后。调用currentTask.getRandomFromLine()方法获取到fromline，

调用currentTask.getRandomSujectLine()获取到Subjectline，处理subjectline的变量，

调用currentTask.getRandomCreative()获取到邮件内容，处理邮件内容的变量。

然后调用相关类库发送邮件。等待返回发送结果，把发送结果提交给任务调度器，由任务调度器写入相应的文件。

账号使用结果发送账号调度器，由它记录使用情况。代理使用结果发送给代理调度器,由它记录使用情况。

currentQueue.currentQuere.currentIndex++

如果收到currentTask暂停或停止请求，则吧curentTask交还给任务调度器(turnback(currentTask))，推出循环。

}

}

如果当前队列未发送完毕，则把currentTask还给任务调度器（turnback(currentTask)）。

程序关闭，先发送任务停止的命令，然所有发送线程结束。